Εισαγωγή στην Πληροφορική & στον Προγραμματισμό

Αρχές Προγραμματισμού Η/Υ (με τη γλώσσα C)

Διάλεξη #8

Παναγιώτης Παύλου

intro-hy-24@allos.gr

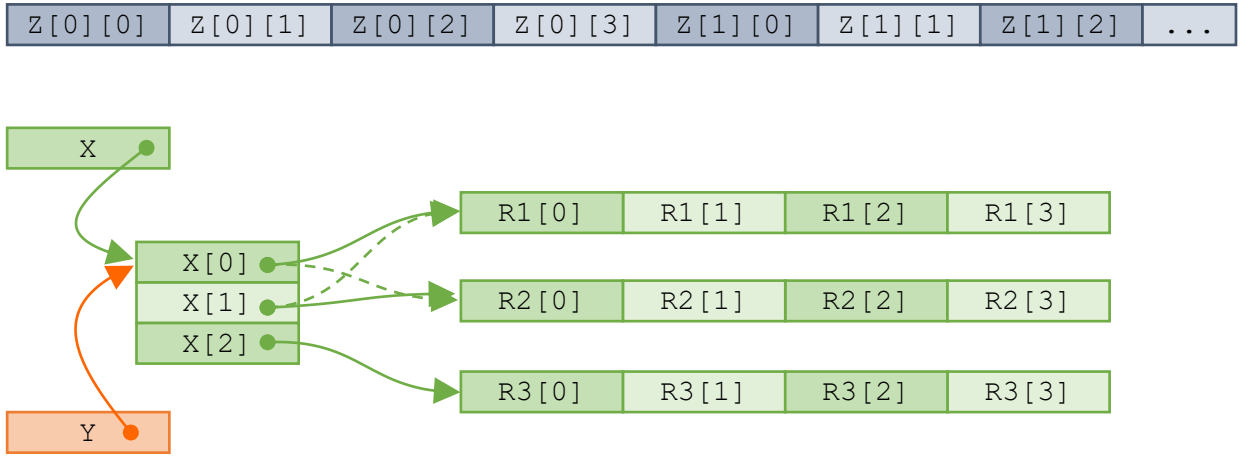
Εφαρμογές δεικτών - 2ο μέρος

Ποιός ο λόγος που μας ενδιαφέρουν οι δείκτες;

Πολυδιάστατοι πίνακες και pointers

Η δυνατότητα να γραφτούν δείκτες σε άλλους δείκτες, αλλά και ο δυϊσμός πίνακα και δείκτη, μας επιτρέπουν να γραφτεί ο δίπλα κώδικας.

Στο διπλό βρόχο στο κάτω μέρος οι X, Y και Z μπορούν να χρησιμοποιηθούν εναλλάξιμα, όμως η δήλωσή τους και η εσωτερική τους δομή δεν είναι η ίδια!




```
double Z[3][4] = {
    { 1,2,3,4 },
    { 5,6,7,8 },
    { 9,10,11,12 }
};

double R1[] = { 1.,2.,3.,4. };
double R2[] = { 5.,6.,7.,8. };
double R3[] = { 9.,10.,11.,12. };
double *X[] = { R1, R2, R3 };
double **Y = X;

for (int row = 0; row < 3; ++row) {
    for (int col = 0; col < 4; ++col) {
        printf("%5.2lf ", Z[row][col]);
    }
    printf("\n");
}
```

Εφαρμογή

Μία συγκεντρωτική εφαρμογή όλων των παραπάνω είναι η δημιουργία ενός constructor και ενός destructor για αλγεβρικούς πίνακες.

- Δημιουργήστε μία συνάρτηση που να επιστρέφει έναν μηδενικό αλγεβρικό πίνακα διαστάσεων m x n
- Δημιουργήστε μία συνάρτηση που να απελευθερώνει τη μνήμη που καταλαμβάνει ένας τέτοιος πίνακας
- Δημιουργήστε μία συνάρτηση που να επιστρέφει τον μοναδιαίο αλγεβρικό πίνακα διάστασης n

Ερωτήσεις?

- Διαβάστε τις σημειώσεις, διαβάστε τις διαφάνειες και δείτε τα videos **πριν** ρωτήσετε
- **Συμβουλευτείτε** τη σελίδα ερωταποκρίσεων του μαθήματος

  https://qna.c-programming.allos.gr
- **Στείλτε** τις ερωτήσεις σας πριν και μετά το μάθημα στο

  intro-hy-24@allos.gr
- Εάν έχετε **πρόβλημα** με κάποιο κώδικα στείλτε τον κώδικα ως κείμενο με copy/paste. Εάν θεωρείτε ότι επιπλέον βοηθά και ένα στιγμιότυπο οθόνης, είναι καλοδεχούμενο.
- Επαναλαμβάνουμε : Μην στείλετε ποτέ κώδικα ως εικόνα μας είναι παντελώς άχρηστος!

Αρχεία

Τα αρχεία ως πόροι – Μοντέλο περιεχομένων – Ειδικά αρχεία

Μοντελοποίηση περιεχομένων 1/2

Τα αρχεία αποθηκεύονται σε μνήμη μακροπρόθεσμης αποθήκευσης (π.χ. σε δίσκο) και είναι ακολουθίες από bytes, κατ'αναλογία με τη μνήμη RAM. Τα bytes είναι και πάλι αριθμημένα, αλλά για κάθε αρχείο ξεχωριστά, οπότε η αρίθμηση του πρώτου byte είναι πάντα μηδέν. Δηλαδή στην αρίθμηση μοιάζουν περισσότερο με τους δείκτες των πινάκων.

Στα αρχεία μεταφέρονται περιεχόμενα από και προς τη μνήμη.

Μοντελοποίηση περιεχομένων 2/2

Παρόλα αυτά διαφέρουν ως προς τον τρόπο της μετακίνησης των δεδομένων από και προς αυτά. Κάθε φορά που ξεκινά η χρήση ενός αρχείου (με το άνοιγμα) δημιουργείται γι' αυτή τη χρήση ένας **δρομέας**. Ο δρομέας αυτός υποδεικνύει το σημείο του αρχείου στο οποίο θα γίνει η **επόμενη** ανάγνωση δεδομένων από το αρχείο ή εγγραφή δεδομένων στο αρχείο. Η αρχική θέση του δρομέα συνήθως, αλλά όχι πάντα, είναι η μηδενική.

Τα αρχεία ως resources 1/3

Τα αρχεία λειτουργούν ως resources (πόροι), δηλαδή για να χρησιμοποιηθούν θα πρέπει, να ανοιχθεί (**open**) πρώτα και με το αποτέλεσμα του ανοίγματος να γίνει η χρήση του. Στο τέλος της χρήσης του να κλειστεί (**close**). Οι εντολές που σχετίζονται με τα αρχεία ξεκινούν με το πρόθεμα f. Όλες περιλαμβάνονται στο stdio.h

Το άνοιγμα γίνεται με την εντολή fopen που συντάσσεται ως εξής:

```
FILE *fopen(char *filename , char *mode)
```

Αυτή είτε επιστρέφει έναν pointer σε μία δομή `FILE`, είτε – ως ένδειξη λάθους – την τιμή `NULL`.

Η πρώτη παράμετρος είναι το όνομα του αρχείου (αν και στην πραγματικότητα είναι η διαδρομή – το path – του αρχείου) ενώ η δεύτερη είναι ένα κείμενο με ένα έως τρεις χαρακτήρες.

Τα αρχεία ως resources 2/3

Το mode παριστάνει τον σκοπό για τον οποίο ανοίγει το αρχείο βάσει του παρακάτω πίνακα:

mode	Χρήση του αρχείου	Το αρχείο υπάρχει ήδη; Ναι	Όχι	Περιεχόμενο	
`r`	Ανάγνωση (read)	✓	Προκύπτει σφάλμα	Κείμενο	
`w`	Εγγραφή (write)	Διαγράφονται τα περιεχόμενα	Δημιουργείται	Κείμενο	
`a`	Προσάρτηση (append)	Προστίθενται στα υπάρχοντα	Δημιουργείται	Κείμενο	*ή binary*
`r+`	Εγγραφή και ανάγνωση	✓	Προκύπτει σφάλμα	Κείμενο	
`w+`	Εγγραφή και ανάγνωση	✓	✓	Κείμενο	
`a+`	Προσάρτηση και ανάγνωση	✓	✓	Κείμενο	

`b`

Τα αρχεία ως resources 3/3

Το κλείσιμο ενός αρχείου γίνεται με την εντολή `fclose` που συντάσσεται ως εξής:

```
int fclose( FILE *handle )
```

Αυτή είτε επιστρέφει 0 εάν όλα είναι εντάξει, είτε μια ειδική τιμή την **`EOF`** που θα συναντήσουμε και αργότερα εάν υπάρχει πρόβλημα. Πρακτικά όμως σπάνια έχει νόημα να γίνει έλεγχος αυτής της τιμής.

Stdin/Stdout/Stderr

Εκτός από τα αρχεία που μπορεί να ανοίξουν με την `fopen` από τον κώδικα, υπάρχουν και τρία ειδικά αρχεία τα οποία είναι ανοιχτά κατά την έναρξη του κώδικα. Αυτά είναι:

`stdin` το αρχείο αυτό αντιπροσωπεύει ό,τι δίνεται ως είσοδος στο πρόγραμμα, από το πληκτρολόγιο

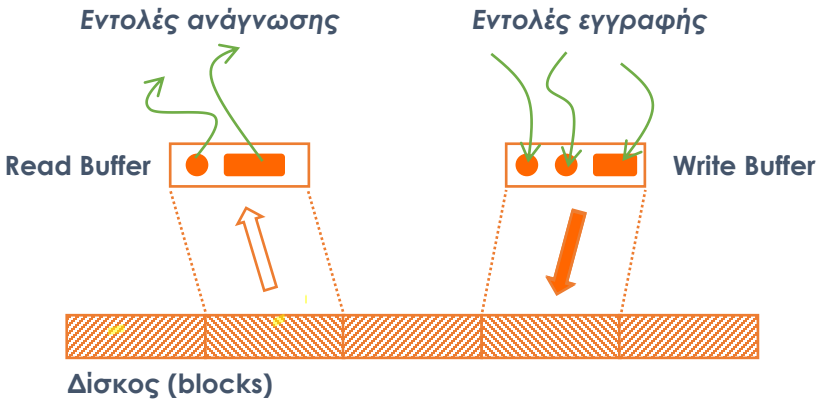
`stdout` το αρχείο αυτό αντιπροσωπεύει ότι δίνει ως έξοδο το πρόγραμμα στην οθόνη αλλά ως αποτέλεσμα της φυσιολογικής λειτουργίας του κώδικα

`stderr` το αρχείο αυτό αντιπροσωπεύει πάλι την έξοδο από τον κώδικα προς την οθόνη, αλλά μόνο τα μηνύματα λάθους προς τον χρήστη

Τα λειτουργικά συστήματα παρέχουν τρόπους όπου τα αρχεία αυτά μπορούν να ανακατευθυνθούν σε άλλες πηγές (`stdin`) ή έξοδος (`stdout`,`stderr`), αλλά αυτό δεν διαφοροποιεί τον κώδικα που γράφεται. Είναι απλά μία δυνατότητα που δίνει το λειτουργικό σύστημα στον χρήστη που εκτελεί ένα πρόγραμμα.

Είσοδος/Έξοδος με buffer

Οι δίσκοι ανήκουν στα block storage devices, δηλαδή σε συσκευές αποθήκευσης όπου η πληροφορία (από/προς) μετακινείται σε τμήματα πληροφορίας προκαθορισμένου μεγέθους (τα blocks), τυπικά μεγέθους μερικών kB.



Αυτό δεν σημαίνει ότι ο κώδικάς μας πρέπει να διαβάζει όμως πληροφορίες ανά block. Αυτή η αποσύζευξη επιτυγχάνεται με τη χρήση ενδιάμεσης μνήμης από το σύστημα, από την οποία ο κώδικας μας διαβάζει ή γράφει και όταν συμπληρωθεί το block, τότε αυτό συγχρονίζεται με τον δίσκο.

Αυτό δημιουργεί πρόβλημα ενίοτε, καθώς τα περιεχόμενα ενός αρχείου μπορεί να μην ευσταθούν, οπότε (όταν χρειάζεται) θα πρέπει να μπορεί ο κώδικάς να ζητά την άμεση μεταφορά των περιεχομένων του buffer προς τον δίσκο.

Αυτό γίνεται με την εντολή:

```
int fflush( FILE *handle )
```

η οποία επιστρέφει αποτέλεσμα όπως και η `fclose`

Ερωτήσεις?

- Διαβάστε τις σημειώσεις, διαβάστε τις διαφάνειες και δείτε τα videos **πριν** ρωτήσετε
- **Συμβουλευτείτε** τη σελίδα ερωταποκρίσεων του μαθήματος

  https://qna.c-programming.allos.gr
- **Στείλτε** τις ερωτήσεις σας πριν και μετά το μάθημα στο

  intro-hy-24@allos.gr
- Εάν έχετε **πρόβλημα** με κάποιο κώδικα στείλτε τον κώδικα ως κείμενο με copy/paste. Εάν θεωρείτε ότι επιπλέον βοηθά και ένα στιγμιότυπο οθόνης, είναι καλοδεχούμενο.
- Επαναλαμβάνουμε : Μην στείλετε ποτέ κώδικα ως εικόνα μας είναι παντελώς άχρηστος!

Αρχεία Κειμένου

Ανάγνωση και εγγραφή κειμένου από και προς αρχεία κειμένου

Μορφοποιημένη Έξοδος

Η μορφοποιημένη έξοδος προς κάποιο αρχείο γίνεται με την εντολή `fprintf` η οποία λειτουργεί ακριβώς όπως η `printf`, αλλά με ένα πρόσθετο όρισμα στην αρχή: το αρχείο στο οποίο θα γραφτεί το αποτέλεσμα (αντί για την οθόνη).

```
int fprintf( FILE *stream, char *format, ... )
```

Παρατηρήστε ότι η `fprintf(stdout, ...)` ταυτίζεται με την `printf(...)`.

Επίσης – αν και εκτός θέματος – μία άλλη παραλλαγή τους είναι η εντολή:

```
int sprintf( char *string, char *format, ... )
```

στην οποία, αντί το κείμενο που παράγεται να οδηγείται σε αρχείο, οδηγείται στη μνήμη που υποδεικνύει ο pointer `string` . Η δεσμευμένη μνήμη εκεί θα πρέπει να είναι αρκετά μεγάλη ώστε να χωράει το αποτέλεσμα.

Μορφοποιημένη Είσοδος

Η μορφοποιημένη είοδος προς κάποιο αρχείο γίνεται με την εντολή `fscanf` η οποία λειτουργεί ακριβώς όπως η `scanf`, αλλά με ένα πρόσθετο όρισμα στην αρχή: το αρχείο από το οποίο θα διαβαστούν τα δεδομένα (αντί για του πληκτρολογίου).

```
int fscanf( FILE *stream, char *format, ... )
```

Παρατηρήστε ότι η `fscanf(stdin, ...)` ταυτίζεται με την `scanf(...)`.

Επίσης – αν και εκτός θέματος – μία άλλη παραλλαγή τους είναι η εντολή:

```
int sscanf( char *string, const char *format, ... )
```

στην οποία αντί το κείμενο που περιέχει τα δεδομένα να προέρχεται από το αρχείο, προέρχεται από τη μνήμη που υποδεικνύει ο pointer `string`.

Είσοδος/Έξοδος ανά γραμμή

Στα αρχεία κειμένου, εφόσον έχουν ανοιχτεί σε κατάλληλο mode, μπορούν να γραφούν και να διαβαστούν ολόκληρες γραμμές με τις δύο παρακάτω εντολές:

```
char *fgets(char *str, int n, FILE *stream)
```

Αυτή διαβάζει μία ολόκληρη γραμμή με το πολύ `n` χαρακτήρες, μαζί με τον χαρακτήρα αλλαγής γραμμής. Επιστρέφει `NULL` είτε σε περίπτωση λάθους, είτε αν ολοκληρωθεί η ανάγνωση ενός αρχείου. Αλλιώς επιστρέφει την τιμή του `str`. Το κείμενο της γραμμής τοποθετείται στη μνήμη που υποδεικνύει ο pointer `str`, που θα πρέπει να έχει δεσμευμένα τουλάχιστον `n` bytes.

```
int fputs(char *str, FILE *stream)
```

Αυτή γράφει το κείμενο str στο αρχείο, και προσθέτει στο τέλος τον χαρακτήρα αλλαγής γραμμής. Σε περίπτωση λάθους, επιστρέφει την τιμή `EOF`.

Είσοδος/Έξοδος ανά χαρακτήρα 1/2

Στα αρχεία κειμένου, εφόσον έχουν ανοιχτεί σε κατάλληλο mode, μπορούν να γραφούν και να διαβαστούν μεμονωμένοι χαρακτήρες με τις ακόλουθες εντολές:

```
int fgetc(FILE *stream)
```

η οποία επιστρέφει τον επόμενο χαρακτήρα του αρχείου ή `EOF` εάν ο δρομέας έχει φτάσει στο τέλος του αρχείου. Η τιμή που επιστρέφεται είναι `int` επειδή εκτός από τους 256 κωδικούς ενός χαρακτήρα char χρειάζεται να μπορεί να παρασταθεί μία τιμή ακόμα, η `EOF`.

```
int fputc(int char, FILE *stream)
```

η οποία γράφει έναν χαρακτήρα στο αρχείο και επιστρέφει είτε τον χαρακτήρα που έγραψε ή την τιμή `EOF` ως ένδειξη λάθους.

Είσοδος/Έξοδος ανά χαρακτήρα 2/2

Η χρήση του buffer μας επιτρέπει και την ακόλουθη «πολυτέλεια». Με την παρακάτω εντολή μπορεί να «επιστραφεί» ένας χαρακτήρας πίσω στον buffer ενός αρχείου που έχει ανοίξει για ανάγνωση.

```
int ungetc(int c, FILE *stream)
```

Επιστρέφει `EOF` σε περίπτωση σφάλματος ή τον χαρακτήρα `c` στον buffer ανάγνωσης (όχι στο ίδιο το αρχείο) και προσαρμόζει κατάλληλα τον δρομέα, ώστε η επόμενη εντολή ανάγνωσης να συνεχίσει από αυτόν. Δεν απαιτείται να έχει διαβαστεί ο χαρακτήρας (αυτός ή άλλος) προηγουμένως από αυτό το αρχείο.

Επίσης χρειάζεται **προσοχή** ότι εκτός από τις εντολές `fputc` και `fgetc`, υπάρχουν και οι παρόμοιες εντολές `putc` και `getc` τις οποίες όμως δεν πρέπει να χρησιμοποιείτε. Τις τεχνικές λεπτομέρειες μπορείτε να τις διαβάσετε σε αυτό το άρθρο, αλλά δεν μας αφορούν στα πλαίσια του μαθήματός μας.

Ερωτήσεις?

- Διαβάστε τις σημειώσεις, διαβάστε τις διαφάνειες και δείτε τα videos **πριν** ρωτήσετε
- **Συμβουλευτείτε** τη σελίδα ερωταποκρίσεων του μαθήματος

  https://qna.c-programming.allos.gr
- **Στείλτε** τις ερωτήσεις σας πριν και μετά το μάθημα στο

  intro-hy-24@allos.gr
- Εάν έχετε **πρόβλημα** με κάποιο κώδικα στείλτε τον κώδικα ως κείμενο με copy/paste. Εάν θεωρείτε ότι επιπλέον βοηθά και ένα στιγμιότυπο οθόνης, είναι καλοδεχούμενο.
- Επαναλαμβάνουμε : Μην στείλετε ποτέ κώδικα ως εικόνα μας είναι παντελώς άχρηστος!

Εφαρμογή αρχείων

1. Δημιουργείστε μία συνάρτηση που να δέχεται το path ενός αρχείου και να μετράει τις γραμμές του
2. Δημιουργείστε μία συνάρτηση που να δέχεται το path ενός αρχείου και μία λέξη (string) και από αυτό το αρχείο να εμφανίζει στην οθόνη μόνο όσες γραμμές του περιέχουν τη λέξη
3. Δημιουργείστε μία συνάρτηση που να σας επιστρέφει το μέγεθος ενός αρχείου σε bytes
4. Δημιουργείστε μία συνάρτηση που να επιστρέφει έναν πίνακα με τα στατιστικά ενός αρχείου κειμένου ως εξής:
(ανά γράμμα, αριθμό, σύμβολα, λευκούς χαρακτήρες)

Σημαντικά σημεία

Μετά από τη σημερινή διάλεξη θα πρέπει να γνωρίζετε:

- Τη χρήση των πολυδιάστατων πινάκων
- Το σύστημα διαχείρισης μνήμης
- Την έννοια του πόρου ενός συστήματος
- Το μοντέλο των αρχείων (και με τον δρομέα)
- Τη διαφορά αρχείων κειμένου και binary
- Την ανάγνωση και εγγραφή αρχείων κειμένου (ανά γραμμή ή χαρακτήρα)